Encoding and Decoding

At the embedder side, for each mark embedding region j ($1<=j<=M$), K frames are chosen to represent the temporal location of that region. These representative frames are termed as ``poles'' in the terminology used in this section and denoted by $\{\boldsymbol{p}_{jk}\}$, where j (resp. k) corresponds to the mark-embedding region (with respective to the index of the pole inside that region), $1<=j<=M$, $1<=k<=K$. Obviously, the set of $\{\boldsymbol{p}_{jk}\}$ is a subset of $\{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_N\}$. Here, how to choose $\{\boldsymbol{p}_{jk}\}$ given a region j will not be discussed. However, in general, as a rule of thumb, poles should be chosen approximately uniformly distributed inside a mark-embedding region so as to represent that region accurately. Let $\{a_{jk}\}$ be the hash values of $\{\boldsymbol{p}_{jk}\}$, i.e., for all j,k, a_{jk} =h($\boldsymbol{p}_{jk}$). The hash values $\{a_{jk}\}$ are sent as side information to the receiver. In other words, it is assumed that the receiver (or the decoder) has perfect knowledge of $\{a_{jk}\}$.

The hash values $\{a_{jk}\}$ are used to "lock" the receiver to the correct position in the attacked video $\{\boldsymbol{y}_i\}$ for each mark embedding region j. In order to achieve this task, the following process must be considered, where ε and α are user-dependent parameters:

1. Find $\{b_1, b_2, \ldots, b_{NN}\}$, where $b_i = h(\boldsymbol{y}_i)$, $1<=i<=NN$.

2. For each pole $\boldsymbol{p}_{jk}$, form the perceptual similarity sets F_{jk} from $\{\boldsymbol{y}_i\}$, where $F_{jk} = \{ \boldsymbol{y}_i \mid d(b_i,a_{jk})< \alpha, 1<=i<=N\}$.

3. For each mark-embedding region j, form the set G_j, which consists of all "temporally-suitable" K-tuples from the similarity sets F_{jk}: $G_j = \{ (\boldsymbol{g}_{j1}, \boldsymbol{g}_{j2}, \ldots, \boldsymbol{g}_{jK}) \mid |td(\boldsymbol{g}_{jk},\boldsymbol{g}_{j,k+1})-td(\boldsymbol{p}_{jk},\boldsymbol{p}_{j,k+1})|< \varepsilon, \boldsymbol{g}_{jk} \in F_{jk}, 1<=k<K\}$.

4. Find the optimal K-tuple for embedding region j in the sense of perceptual similarity via hash: $(\boldsymbol{g}_{j1}^*, \boldsymbol{g}_{j2}^*, \ldots, \boldsymbol{g}_{jK}^*) = \text{argmin} \sum_{k=1}^{K}$

$d(h(\mathbf{g}_{jk}), a_{jk})$, where the minimization is carried out over all element of G_j.

5. The K-tuple $(\mathbf{g}_{j1}^*, \mathbf{g}_{j2}^*, \ldots, \mathbf{g}_{jK}^*)$ determines the j-th embedding location in $\{\mathbf{y}_i\}$.

Remark

Note that, by using this straightforward process, steps 3 and 4 take $O(K \prod_{k=1}^{K} | F_{jk} |)$ operations. The reason is that the total number of possible K-tuples is $\prod_{k=1}^{K} | F_{jk} |$ (i.e., exponential in K) and for each K-tuple, this approach needs to perform $O(K)$ operations to find its optimal match in the sense of perceptual similarity (in other words, the Hamming distance to the original hash values). However, there is redundancy in these operations because there exist K-tuples that have common elements for which the Hamming distances between the hash values does not need to be recalculated. Thus, a computationally more efficient approach to solve steps 3 and 4 jointly can be applied by using dynamic programming.

Pseudo-Code

The following pseudo-code is presented to illustrate the basic idea by using dynamic programming. This would replace steps 3 and 4 above for any j. Furthermore, let $F_{jk} = \{\mathbf{g}_{jkl}\}$, where l indexes the order of each set element.

I. Initialize *mindist* to a very large number and $k=1$, $l=1$.

II. While $1<=l<=F_{jk}$, do

II.I. Initialize the K-tuple *path* such that $path(m)=0$ if $m \neq k$ and $path(k)=\mathbf{q}_{jkl}$, where $path(k)$ is the k-th entry of *path*.

II.II. Initialize $dist=d(a_{jk}, h(path(k)))$, $VALIDITY=GOOD$.

II.III. Apply function $FINDOPTPATH(path, dist, k+1, l, VALIDITY)$ which is defined below.

II.IV. Increment *I* by *1*, go to step II.I.

function FINDOPTPATH(path,dist,k,I,VALIDITY)

I. Initialize *II=1*.

II. While *II<=|F_{jk}|* do

II.I. Compute *timedist=|td(path(k-1),* $\mathbf{q}_{j,k,II}$*)-td(*$\mathbf{p}_{j,k-1}$,$\mathbf{p}_{jk}$*)|*.

II.II. If *(k<K)* and *(timedist >ε)*,

II.II.I. Set *VALIDITY = BAD*.

II.II.II. Apply function *FINDOPTPATH(path,dist,K,II,VALIDITY)*.

II.III. Else if *(k<K)* and *(timedist<=ε)*,

II.III.I. Set *path(k)=*$\mathbf{q}_{j,k,II}$, and increment *dist* by $d(a_{jk},h(path(k)))$.

II.III.II. Apply function *FINDOPTPATH(path,dist,k+1,II,VALIDITY)*.

II.IV. Else if *(k=K)* and *(dist<mindist)* and *(VALIDITY=GOOD)*, set *mindist=dist* and *minpath=path*.

II.V. Increment *I* by *1*, go to step **II.I.**

The foregoing description of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description of the invention, but rather by the claims appended hereto.